VECTOR GRAPHIC INC. PRESENTS THE SOURCERER

A COMPREHENSIVE DISASSEMBLER FOR 8080 OBJECT CODE: The Sourcerer is designed to convert 8080 object code files to source code files, which are easier to understand and modify. The code is systematically converted to Intel standard mnemonics, and the created file is compatible with most 8080 assemblers, including ESP-1 and others based on the Processor Technology assembler.

CREATE ESP FILES DIRECTLY IN RAM: Files may be formed directly in memory, or may be listed on any output terminal device. Several formats are available, including two with decimal line numbers and formatted constants and symbols. All symbols have a leading H followed by their hex equivalents; thus the assembler treats them as labels. All numerical constants have a leading zero and trailing H so they are treated as hexadecimal constants.

SYMBOL TABLE: The Sourcerer operates in two passes. On the first pass, all three-byte codes are used to form a hex symbol table. This table is used on the second pass to affix labels to the file. The symbol table may be listed separately.

CROSS-REFERENCE TABLE: A cross-reference table may be listed. This table consists of each symbol, the RAM addressed from where it is referenced, and the mnemonic codes that reference it.

The Sourcerer is a unique tool, a must for the true software buff. It allows you, in a very short time, to create source files from all those machine code programs that you want to modify, but are reluctant to hack up with haywire patches. Now you can create your own source file, add commands, change subroutines, or delete unused features. Your newly assembled program will be as efficient and compact as the original!

To use the Sourcerer, you'll need a dedicated 4K block of memory. The code itself occupies just over 2K with another short section devoted to scratchpad, input buffer and stack. The remainder of the 4K block is used for symbol table storage.

VECTOR GRAPHIC INC.

THE SOURCERER

BY

TOM NUSSMEIER

START 0000

LENGTH 0930

FCB 0018 : ASM 1000 Hex

THE SOURCERER

The SOURCERER is a comprehensive disassembler designed to convert 8080 object code files to source code files, which are easier to understand and modify. The code is systematically converted to INTEL standard mnemonics, and the created file is compatible with most 8080 assemblers, including ALS-8, Imsai, ESP-1 and others based on the Processor Technology assembler. To achieve this, the SOURCERER operates in two separate passes, one to form a symbol table and the second to convert the code to mnemonics. The symbols are placed at the beginning of the appropriate lines as the second pass progresses. For maximum versatility, the two passes are controlled separately. For several of the formats, the symbols are simply hex addresses. In those formats intended for use in an assembler, the symbol is preceded by an H, (example - H237F) so that the assembler will interpret them as labels, not numbers.

The SOURCERER forms the symbol table by examining all the codes in a block of memory (specified by the command), collecting the last two bytes of all three-byte codes, sorting these 16 bit addresses in numerical order, and storing them in a file for future use. It is important to note that all these codes are treated as labels, even though some may be constants. During reassembly, the constants may cause errors, but these must be located the hard way; there is no way to distinguish a constant from a label in an object code file. A similar problem occurs with data tables buried in the object code; the SOURCERER will treat these as commands and attempt to decode them. These must be located and output using the D command (see description of commands below) to prevent errors. Finally, the assembler will require some labels that the SOURCERER will not locate. These usually occur when a data table is used for command decoding. It will be necessary after forming the file to locate these areas and type in the required labels.

Despite the limitations listed above, the SOURCERER is a very powerful tool. The author has used it to create source files from object codes as long as 5000 bytes, and has used these files to modify and reassemble programs both at the same address as the original, and at new locations. Typically, a 1000 byte object can be converted to source in one evening's work, starting with very little knowledge of the program. I recommend, however, that the SOURCERER be used on some of your own software, preferably short programs, to get the necessary familiarity with its operation.

COMMAND DESCRIPTION

There are 12 commands, consisting of one letter each. Many of the commands require addresses; these are typed directly following the commands in hexadecimal notation. Commands are executed by the carriage return, and may be edited or aborted up to the time of execution.

Prompt string - when properly loaded, the program will respond with a triangular bracket and space (>). This signifies that the program is awaiting a command.

Entry and editing - Commands are typed in as single letters. If a non-existent command letter is typed, it will be rejected by the program. The rejected letter will be printed followed by an underline character (hex 5F), which will erase on most video terminals. Proper commands will not be erased. If address fields are to be typed following the command, they must be in hex. Any non-hex character will be rejected. Leading zeros are not required. If more than four hex characters are typed, only the last four will be used. Any character may be erased by typing an underline (5F). Address fields are separated by spaces, but the space between the command and the first field is optional. Any command may be aborted by Control-C. Commands will be discussed in the order in which they are most likely to be used.

S BBBB EEEE - Form symbol table. (BBBB and EEEE are the beginning and end of the code block to be accessed). The symbol table is formed and stored on this pass through the code. If there are tables or other non-code entries in the block, these will be interpreted as code and erroneous entries will be made. (Examining the symbol table for unusual addresses is one method for locating tables.)

T - Display symbol table. Outputs symbol table as a series of hex addresses. 0000 is always output; the dissassembler uses this value to locate the beginning of the symbol table. Table is listed sequentially, 8 addresses per line.

X BBBB EEEE - Display cross-reference table. This command lists every symbol in the symbol table followed by the address and mnemonic code of every location that references the symbol. This is very useful when programs are being modified. However, when long programs are involved, this command is very slow, and the user should be aware that long times may be involved. As an example, printing the cross-reference table for a 5K program can take up to 2 hours. (Note: if a third address field is used with this command, the first symbol to be listed will be equal to or higher than this third field.)

F AAAA - Set RAM file location. Initializes the counters for creating files in RAM. If AAAA = 0, no file will be created. If F is typed without any addresses, the starting and ending address of the file will be displayed. Only two commands will cause a RAM file to be built, D (Data Word) and L (List Format). Other commands will not affect the RAM area.

O LLLL RRRR - Set location offset. If the program to be disassembled is not located at it's usual running address, this command will compensate. LLLL is the address where the object code is located; RRRR is the address where the code normally runs.

D BBBB EEEE - Data word format. Outputs code block as Intel
pseudo-op DW. Useful for forming tables for the assembler. This
command and the L command will form files in RAM if the R F
command has been used to set a starting address.

N BBBB EEEE - Mnemonic format. This command lists the program
with hex addresses, the hex version of the code, the ASCII code
equivalent, the label, mnemonic, arguments, and data fields, if
applicable. ASCII equivalents for two and three byte codes are
output on the same line. As with all formats (except L), the
listing will appear with all labels, mnemonics, arguments, etc.,
vertically aligned for easy reading. The N command is
recommended for preliminary decoding of a new program.

P BBBB EEEE - Print format. This command dumps code with
decimal line numbers, labels with leading H, mnemonics,
arguments and data fields. Two-byte data fields will have a
leading H to allow referencing to the labels. One-byte data
fields will have a leading zero and a trailing H, so that the
assembler will treat them as hex constants. This format is one
of two that will load directly into an assembler. This
particular format is useful only with short programs, as it does
add a lot of extra spaces. However, it is very easy to read.

L BBBB EEEE - List format. This command is the same as P except
all the excess spaces have been deleted. This command should be
used when dumping code onto paper tape for reload ing into an
assembler. This command is also the principle command for
creating a file in memory.

R DDDD II - Reset line numbers. This command establishes the
first decimal line number (DDDD) and the line number interval
that will be output with the P, L and D commands. Both DDDD
and II should be decimal numbers. This command initializes to a
starting value of 1 and an interval of 1.

G AAAA - Goto. Transfers program control to the specified
address AAAA. This is a utility command included to allow
operating several programs in memory simultaneously without the
need for transfering with the front panel switches.

I - Initialize. The SOURCERER is self-initializing the first
time it is accessed. After this first initialization, I must be
used to re-initialize. When I is typed, followed by a carriage
return, the terminal will respond with a message which reads:
TYPE Y TO INITIALIZE. At this point, if a Y is typed, full
initialization will occur. Any formed symbol table will be
deleted, the O and F commands will be reset to zero (Offset and
RAM file) and the decimal number counter will be reset to it's
initial value. If any letter other than Y is typed, only the
decimal counter will be reset. (The message is output to
prevent inadvertent loss of the symbol table.)

LOADING AND PATCHING THE SOURCERER

The SOURCERER requires a dedicated 4K block of memory. The code itself occupies just over 2K, with another short section devoted to scratchpad, input buffer and stack. The remainder of the 4K block is used fo symbol table storage. The table is built from the top down, thus even the smallest table will use the zone at the end of the 4K block. There is room in the table for about 700 symbols, which is adequate for nearly any program you might want to disassemble. (A 12K object code will have just about 700 symbols).

+ 2302

I/O PATCHING

The SOURCERER is delivered with standard TTY I/O routines because this is a starting point that most people are familiar with. The output routine is located at Hex X0B0. (The X is the first number of the version that you are using; 3 for the 3000H version, 0 for the 0000 version). Four NOP's follow this routine; they are available for use if necessary. The input routine is located immediately following these NOP's at Hex X0C0. Again, four extra bytes are available after this routine. Be sure that your output routine does not modify any of the registers, and that the input character is returned in the A register.

Immediately following the input routine is the pause routine. This routine allows dumps or listings to be interrupted by the spacebar or aborted by control-C. At location X0D2 the keyboard data port is read (input). This location is delivered with hex code 01, (the TTY data port); change it to match your keyboard port. This completes I/O patching. The input/output and pause routines are listed on the next page.

SOURCERER EXAMPLE

A short program, designed to run at 3000 hex, has been loaded into memory at location 1000. The SOURCERER is loaded at 3000, thus an offset is required. A file of the program will be formed at hex 4000. The program code is 32 hex bytes long. Upper case letters represent the output when running the SOURCERER; lower case comments were added later for clarity.

Hex Dump of program.

```
1000:   21 00 00 CD 22 30 DB FF E6 40 CA 13 30 CD 03 F0
1010:   C3 03 30 DB FF E6 80 C2 2C 30 CD 03 F0 77 23 C3
1020:   03 30 DB 27 E6 08 C2 22 30 DB 26 C9 3E 5F CD 03
1030:   F0 C9
```

```
> I                                    Initialize Command.  Not re-
TYPE Y TO INITIALIZE                   quired   when first starting.


> S 1000 1031                          Form Symbol Table.


> T                                    Display Symbol Table.

SYMBOL TABLE

0000      3003      3013      3022      302C      F003


> X 1000 1031                          Display Cross-Reference Table.

0000
1000 LXI H

3003
1010 JMP      101F JMP

3013
100A JZ

3022
1003 CALL     1026 JNZ

302C
1017 JNZ

F003
100D CALL     101A CALL     102E CALL


> F 4000                               Set File Starting Address.

FILE  4000  4000


> O 1000 3000                          Set  offset - Program Runs at
                                       Address 3000 Hex


> N 1000 1031                          Use N Command for First Pass.

1000:   21   !                LXI    H,0000     Note Hex Addresses,
1003:   CD   3"0   3003       CALL   3022       Hex   Codes,  ASCII.
1006:   DB                    IN     FF         Labels are  Offset.
1008:   E6   @                ANI    40         No Leading  0's  or
100A:   CA   0                JZ     3013       H's for this Format.
100D:   CD                    CALL   F003
1010:   C3   0                JMP    3003
1013:   DB         3013       IN     FF
1015:   E6                    ANI    80
```

```
1017:   C2    ,0           JNZ    302C
101A:   CD                 CALL   F003
101D:   77    w            MOV    M,A
101E:   23    #            INX    H
101F:   C3    0            JMP    3003
1022:   DB    '     3022   IN     27
1024:   E6                 ANI    08
1026:   C2    "0           JNZ    3022
1029:   DB    &            IN     26
102B:   C9                 RET
102C:   3E    >     302C   MVI    A,5F
102E:   CD                 CALL   F003
1031:   C9                 RET
```

> F   Examine File Length;   File
                                    not Formed by "N" Command.

FILE 4000   4000

R 0 10   Set Decimal Line Numbers to
                                    Start at Zero, Count by 10.

> L 1000 1031   List Format. This Format has
                                    Decimal Line Numbers, Proper
```
0000   LXI H,H0000
0010 H3003 CALL H3022
0020   IN 0FFH
0030   ANI 040H
0040   JZ H3013
0050   CALL HF003
0060   JMP H3003
0070 H3013 IN 0FFH
0080   ANI 080H
0090   JNZ H302C
0100   CALL HF003
0110   MOV M,A
0120   INX H
0130   JMP H3003
0140 H3022 IN 027H
0150   ANI 008H
0160   JNZ H3022
0170   IN 026H
0180   RET
0190 H302C MVI A,05FH
0200   CALL HF003
0210   RET
```
                                    Spacing, Leading H, to form
                                    Label From Hex Address Symbols.
                                    Constants Have Leading 0 and
                                    Trailing H. This File Also
                                    Formed in RAM at Hex 4000.

> F   Examine File Length

FILE 4000   4175

> D 1000 1008   Dump a Little in DW Format.

```
0220    DW  00021H
0230    DW  0CD00H
0240    DW  03022H
0250    DW  0FFDBH
0260    DW  040E6H


> R 1 1                              Reset Decimal Line Numbers.


> P 1000 1008                        Use the Print Format.

0001            LXI     H,H0000
0002 H3003      CALL    H3022
0003            IN      0FFH
0004            ANI     040H
>
```

```
00B0  F5                   0735  CRT1    PUSH    PSW         OUTPUT ROUTINE
00B1  DB  00               0740          IN      00
00B3  E6  80               0745          ANI     80H
00B5  C2  B1  00           0750          JNZ     CRT1+1
00B8  F1                   0755          POP     PSW
00B9  D3  01               0760          OUT     01
00BB  C9                   0765          RET
00BC                       0770  ::                          4 BYTES RESERVED
00BC                       0775          DS      4           FOR OUTPUT ROUTINE
00C0                       0780  ::
00C0  DB  00               0785  KEYB    IN      00          INPUT ROUTINE
00C2  E6  01               0790          ANI     01
00C4  C2  C0  00           0795          JNZ     KEYB
00C7  DB  01               0800          IN      01
00C9  E6  7F               0805          ANI     7FH
00CB  C9                   0810          RET
00CC                       0815  ::                          4 BYTES RESERVED
00CC                       0820          DS      4           FOR INPUT ROUTINE
00D0                       0825  ::
00D0  F5                   0830  PAUSE   PUSH    PSW         PAUSE ROUTINE
00D1  DB  01               0835          IN      01          READ DATA PORT
00D3  E6  7F               0840          ANI     7FH
```